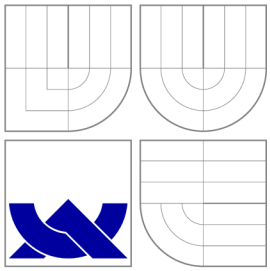
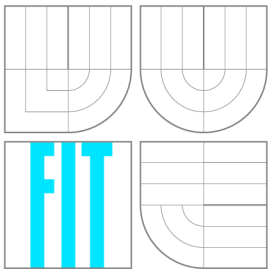


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE 3D OBJEKTŮ REPREZENTOVANÝCH MNOŽINOU POVRCHOVÝCH BODŮ

3D OBJECT VIZUALIZATION REPRESENTED BY BOUNDARY POINTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUDĚK HEZINA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PŘEMYSL KRŠEK, Ph.D.

BRNO 2007

Zadání

1. Seznamte se s problematikou objektů reprezentovaných množinou povrchových bodů
2. Analyzujte problematiku zobrazování objektů reprezentovaných množinou povrchových bodů
3. Navrhněte systém pro 3D zobrazování objektů reprezentovaných množinou povrchových bodů
4. Implementujte a otestujte navržený systém
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu

Licenční smlouva

Licenční smlouva je uložena v archíve Fakulty Informačních Technologií Vysokého Učení Technického v Brně.

Abstrakt

Cílem této práce bylo vytvoření systému pro vizualizaci 3D objektů reprezentovaných množinou povrchových bodů. Práce je napsána v jazyce C++. Na vytvořeni okna, manipulaci se scénou, a načítání objektů z externích souborů jsem využil knihovnu OpenSceneGraph. Do systému jsou zahrnuty dva integrované generátory implicitních objektů.

Klíčová slova

programovací jazyk C++, OpenGL, OpenSceneGraph, množina povrchových bodů, bodově orientované zobrazení

Abstract

The aim of my bachelor thesis was to create the system for visualization of 3D objects which are representated by boundary points. Thesis is written in C++ language. I have used the library OpenSceneGraph for creation of windows, manipulation with scene and for loading objects from external files. The system includes two integrated generators of implicit objects.

Keywords

programing language C++, OpenGL, OpenSceneGraph, Boundary Points, Point-based rendering

Citace

Luděk Hezina: Vizualizace 3D objektů reprezentovaných množinou povrchových bodů, bakalářská práce, Brno, FIT VUT v Brně, 2007

Vizualizace 3D objektů reprezentovaných množinou povrchových bodů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Přemysla Krška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Luděk Hezina
15. května 2007

Obsah

Kapitola 1

Úvod

První užití 3D diskrétního vzorku jako vykreslovaného primitiva v počítačové grafice se datuje do roku 1983, kdy vznikl pojem particle system (částicový systém). Částice je jednoduše bod ve 3D Euklidovském prostoru, kombinovaný s vlastnostmi jako barva, hustota, stínování a rozptylový koeficient. Hlavní výhodou částicového systému je, že vykreslovací krok se stává triviálním. Pro každou částici se provede: projekce na obrazovku, zjištění viditelnosti pomocí Z–bufferu a následné vybarvení pixelu barvou uchovávanou v částici. Částicové systémy se především užívají jako modelovací primitivum k generování specifických animovaných úkazů, které se obtížněji popisují klasickými modelovacími technikami, jako je oheň, déšť či exploze.

Taktéž je možné využít částicového systému k popisu plochy libovolného 3D modelu. Při tomto použití každá částice reprezentuje kousek plochy objektu, ležícího pod ním, a přejímá její vlastnosti (normálový vektor, barvu). Orientován ve v ploše tečné k bodu, ležícímu na povrchu plochy popisovaného objektu.

Orientovaný bod v prostoru, jenž má velikost, se v počítačové grafice nazývá surfel (z anglického *surf*face *el*elemt [5]).

V následujícím textu se budu zabývat různými metodami reprezentace 3D modelů v počítačové grafice, následovaném rozborem surfelu, návrhu celého systému, jeho implementací. Nakonec zhodnocením výsledků celé práce a navržení možného dalšího vývoje.

Kapitola 2

Analýza problému

2.1 Reprezentace 3D modelů

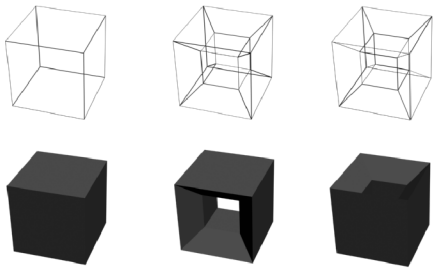
V počítačové grafice se využívá několik druhů reprezentací 3D modelů objektů. A to jak pro popis povrchu, tak pro popis objemu.

2.1.1 Hraniční reprezentace

Hraniční reprezentace neboli B–rep (z anglického Boundary representation) popisuje 3D objekt pomocí jeho povrchu (hranice). Informace o vnitřku tělesa se neukládají.

- Drátový model

  Drátový model se skládá pouze z bodů a hran (úsečky mezi body). Tato reprezentace má nedostatečné množství topologických informací, což může vést k nesprávné interpretaci objektu.

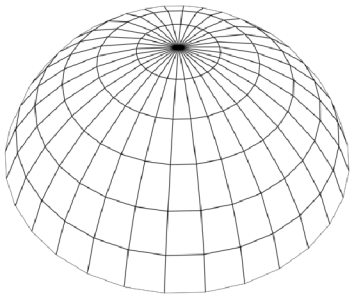


Obrázek 2.1: Ukázka drátových modelů (nahoře) a s ním korespondujících skutečných tvarů

  Jak je vidět na obrázku 2.1 nedostatek topologických informací vede u prostředního a pravého objektu k více než jedné interpretaci. Proto se tato technika hodí převážně pro rychlé orientační zobrazení objektů, například při práci se složitými 3D modely.

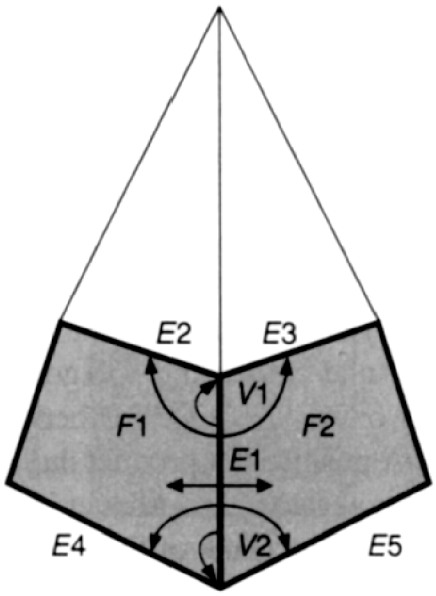
- Polygonální síť

  Slouží k popisu topologie 3D modelu, kde celý povrch modelu je reprezentován polygony a s tím souvisejícími hranami a vrcholovými body. Množina vrcholových bodů je pospojována pomocí hran tak, že vytváří souvislou síť polygonů 2.2.



Obrázek 2.2: Síť polygonů tvořící polokouli

Obvykle bývá tato síť reprezentována datovou strukturou zvanou okřídlená hrana. Kdy každou hranu svírají maximálně dva polygony. Nese informaci o vrcholech, mezi kterými leží, o polygonech, které ji svírají, a informace o dvou dalších hranách vystupujících z každého vrcholu náležejícího hraně (viz. obrázek 2.3). Z toho plynou tři lineární seznamy, a to seznam vrcholů, hran a stěn.



Obrázek 2.3: Znázornění informací, jenž přenáší hrana [3]

- NURBS plocha

  NURBS (non–uniform, rational B–spline) je matematický model užívaný pro generování a reprezentaci křivek a povrchů. NURBS křivka je v pořadí definována váhovými koeficienty řídících bodů a uzlovým vektorem. NURBS křivky a plochy jsou zobecněním nad B–spline křivkami. Pro průchod NURBS křivkou se využívá jediného parametru, obyčejně nazývaného s nebo u, kdežto u ploch jsou tyto parametry dva (s a t nebo u a v). Plocha je definována

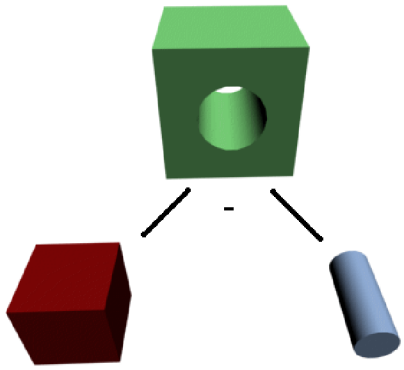
$S(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}}$,
kde $N_{i,p}$ a $N_{j,q}$ jsou bázové funkce B–spline křivky, $P_{i,j}$ kontrolní body a $w_{i,j}$ je váha bodu $P_{i,j}$ [4].

- Popis povrchu body

  Množina bodů rozmístěných po povrchu libovolného objektu, jenž splňuje vzorkovací kritéria plně popisuje geometrii a topologii tělesa. Jednotlivý bod obsahuje informaci o své pozici, orientaci (normálový vektor) a barvě. Dále se dá předpokládat že zabírá jistý prostor a má kruhovitý či elipsovitý tvar. Pro korektní zobrazení musí body pokrývat celý povrch bez děr, z čehož plyne nutnost překrývání [1].

2.1.2 CSG strom

CSG (constructive solid geometry) je technika užívaná při modelování. Využívá jednoduchých primitiv jako krychle, koule, čtyřstěn či válec a booleovských operací (spojení $\cup$, průnik $\cap$ a rozdíl $\setminus$) k jejich kombinování do výsledného tvaru.



Obrázek 2.4: Rozdíl dvou primitiv

Pomocí 3D primitiv transformací a booleovských operací (uzly stromu) je následně možné vytvářet složitější modely. Jelikož v SCG stromu nejsou informace o povrchu objektu převádí se na model polygonální.

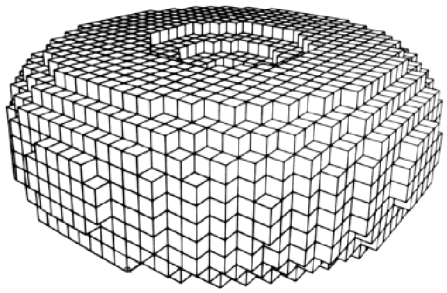
2.1.3 Dekompoziční modely

Dekompoziční modely popisují objekt jeho rozložením do buněk (voxelů) uspořádaných do pevné pravidelné mřížky. Pro reprezentaci objektu je nutné rozhodnout, které buňky v mřížce jsou obsazeny. Využívají se pro popis objemových dat [3].

Metody uložení dat:

- 3D pole diskrétních hodnot

  Vytočí se trojrozměrné pole nad nejmenším kvádrem opsaným popisovaného objektu, a následně se vyplní podle obsazenosti. Vhodné pro vysokou rychlost přístupu, ale přináší velkou paměťovou náročnost.

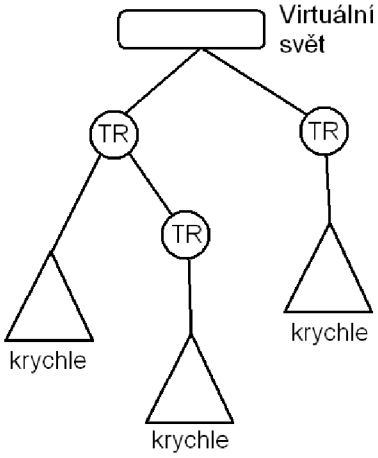


Obrázek 2.5: Objekt složený z buněk [3]

- Oktanový strom Je odvozen od 2D varianty kvadrantového stromu, založeném na binárním dělení. Rekurzivně dělí prostor kolem objektů do osmi částí. Každá takto vytvořená část může nabývat pouze třech stavů, a to: plně obsazen, částečně obsazen a neobsazen. Nadále se rozdělí každá částečně obsazená oblast. V dělení se pokračuje do té doby, pokud nejsou kvadranty homogenní, nebo se nedosáhlo určené hloubky zanoření (minimální prostorové velikosti buňky). Tato varianta je výhodná pro malou hustotu dat, za to přináší problematické procházení stromem.

2.2 Graf scény

Graf scény je reprezentován stromem. Jedná se o datovou strukturu reprezentující celou scénu (virtuální svět), jak 3D modely v něm obsažené, jejich prostorové transformace, rotace popřípadě animace či jiné nastavení stavů. Typicky se schématicky kreslí s kořenovým uzlem nahoře a větvemi směrem dolů. Začíná kořenovým uzlem reprezentujícím cely virtuální svět, ať již 2D, či 3D. Následně je virtuální svět rozdroben do hierarchie uzlů obsahující, informace o pozici, natočení, animaci, či definici logických vztahů mezi objekty, jako například ovládání stavů semaforu (zelená, červená). Následně ve větvích stromu jsou obsaženy samotné vykreslitelné objekty s příslušným nastavením materiálů.



Obrázek 2.6: Graf scény

Na obrázky 2.6 je příklad grafu scény reprezentující tři krychle. Z nichž dvě sdílí jeden shodný modifikátor prostorové transformace TR. Následně se graf rozděluje na samotnou krychli a další transformaci krychle druhé. V pravé části stromu se nachází pouze jedna transformace k ní náležící krychle.

2.3 Splat jako vykreslovací primitivum

Splat je rovinný útvar určený normálovým vektorem $\vec{N} = (N_x, N_y, N_z)$ kde $N_x, N_y, N_z \in R$, středovým bodem $P = (P_x, P_y, P_z)$ pro $P_x, P_y, P_z \in R$ a velikostí v. Pro jeho vytvoření je nejprve nutné v rovině určené normálovým vektorem $\vec{N}$ a bodem P zkonstruovat trojúhelník, nejlépe rovnostranný, kolem bodu P tak, aby bod P ležel v těžišti trojúhelníku. Pro konstrukci trojúhelníku v ploše je nutné znát dva na sebe kolmé vektory v rovině konstrukce. Tento výpočet je udán rovnicemi 2.1 a 2.2.

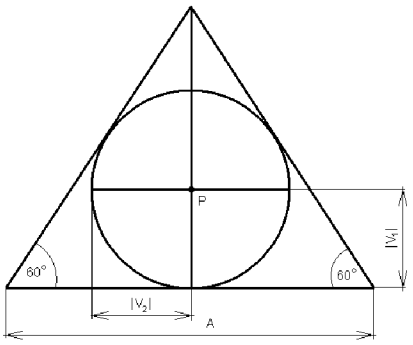
$$\vec{V_1} = (-N_z - N_y, -N_z - N_x, N_y + N_x) \quad (2.1)$$

$$\vec{V_2} = (V_{2_x}, V_{2_y}, V_{2_z}) \quad (2.2)$$

$$V_{2_x} = -N_z * N_z + N_x * N_z - N_y * N_y - N_y * N_x$$

$$V_{2_y} = N_z * N_z + N_y * N_z + N_y * N_x + N_x * N_x$$

$$V_{2_z} = -N_z * N_y - N_y * N_y + N_z * N_x - N_x * N_x$$



Obrázek 2.7: Nákres rovnostranného trojúhelníku zkonstruovaného kolem bodu P

Kružnice na obrázku 2.7 reprezentuje výsledný splat, o velikosti $v = |\vec{V_1}| = |\vec{V_2}|$. Pak jednotlivé vrcholy trojúhelníku reprezentují rovnice 2.3, 2.4 a 2.5.

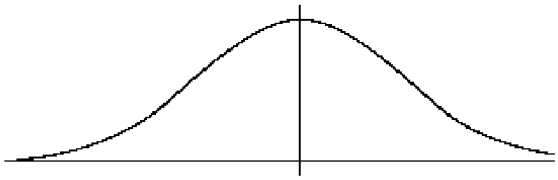
$$B_1 = 2 * \vec{V_1} + P \quad (2.3)$$

$$B_2 = -1 * \vec{V_1} + \sqrt[2]{3} * \vec{V_2} + P \quad (2.4)$$

$$B_3 = -1 * \vec{V_1} - \sqrt[2]{3} * \vec{V_2} + P \quad (2.5)$$

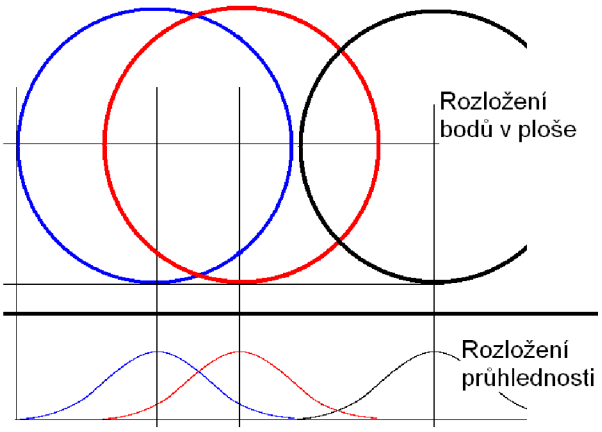
Trojúhelník složený z takto vypočtených bodů B_1, B_2 a B_3 je rovnostranný a leží v rovině reprezentované bodem P a normálovým vektorem $\vec{N}$. V další fázi se rovnostranný trojúhelník otexturuje tak, že výsledný tvar bude kulatý. Aby bylo možné zobrazovat poloprůhledné objekty bez výrazných přechodů překrývajících se splatů je výhodné použít

kruhovou texturu s rozložením průhlednosti dle obrázku 2.8, kde nejvyšší hodnoty rozložení představují neprůhledné části textury.



Obrázek 2.8: Gaussovo rozložení

Při reprezentaci povrchu objektů pomocí takto vytvořených entit je potřeba dbát na jejich rozložení po povrchu modelu, tak aby nedocházelo k jejich přílišnému překrývání, z důvodu nekonstantní míry průhlednosti.



Obrázek 2.9: Vztah mezi rozložením průhlednosti a vzdálenostmi bodů

Z obrázku 2.9 je patrné, že při zvýšení vzdálenosti středů bodů se v místě překrývání podstatně mění celková velikost průhlednosti pro danou oblast.

2.4 Požadavky na systém

Aby byl systém prakticky využitelný plynou z toho následující požadavky:

- schopnost rychlého zobrazení velkého objemu dat
- zobrazení většího počtu objektů do sebe
- nastavitelnost průhlednosti u libovolného objektu
- nastavení velikosti surfelu
- manipulace se scénou
- generování testovacích objektů

- načítání objektů z externího zdroje
- převod polygonálního modelu

Schopnost rychlého zobrazení velkého objemu dat je základním stavebním kamenem celého systému. Uvažujeme–li zobrazování stovek tisíc bodů, né–li více je nutné zaručit v dané situaci co možná nejvyšší rychlost překreslovaní celé scény. Pro vyšší využitelnost systému je vhodné mít možnost komponovat scénu z více než jednoho 3D modelu. S tím souvisí i nutnost takto zkomponovanou scénu korektně zobrazit.

Při komponování scény z více než jednoho 3D modelu se nezřídka stane, že tělesa umístěné uvnitř jiných těles nebudou vidět. Což sice není chyba, ale je vhodné vnějším tělesům nastavit jistou míru průhlednosti, tak aby všechny zobrazované tělesa byly zřetelné.

Jelikož se 3D modely obvykle liší svou velikostí co do délky, výšky či šířky není vždy nutné popisovat jejich povrch stejně velikými surfely. Rozměrově náročnější trojrozměrný útvar, s nevýraznými nerovnostmi povrchu je výhodné aproximovat surfely vyšších rozměrů, než objekt, který co do velikosti je nepatrný, ale za to má členitý povrch.

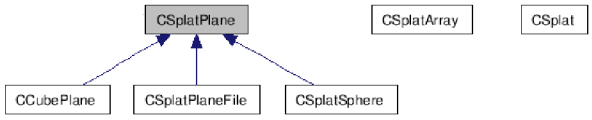
Z hlediska využitelnosti systému je výhodné do něj začlenit manipulaci se scénou. Jak rotace kolem všech os tak, i posunutí do libovolného bodu. Pro otestování funkčnosti zobrazení i manipulace se scénou je nutné vytvořit sadu testovacích 3D modelů, které budou přímo integrovány do systému.

Na rozdíl od testovacích objektů zahrnutých ve vnitřní struktuře systému, je vhodné připojit část, která bude mít za úkol načítání 3D modelů z externího zdroje. Jelikož veliké množství takto načtených 3D modelů bude pravděpodobně reprezentováno polygonální sítí, je nutné takto tvořený trojrozměrný útvar převést na útvar tvořený body.

Kapitola 3

Návrh systému

V této kapitole se budu zaobírat rozdělením do jednotlivých funkčních celků a jejich napojení do zbytku systému. Systém je postaven nad knihovnou OpenSceneGraph. To předpokládá existenci hlavního vykreslitelného objektu (CSplatPlane), který je připojitelný do grafu scény OSG. K tomu slouží čistě virtuální třída OSG::Drawable, ze které bude samostatně vykreslitelný objekt odvozen. Tato třída po připojení do grafu scény zaručí, že při průchodu grafu ve vykreslovací části se provede virtuální metoda drawImplementation, která provede vykreslení celého modelu.



Obrázek 3.1: Diagram tříd

3.1 Samostatně vykreslitelná entita

Samostatně vykreslitelná entita je reprezentována třídou CSplat, jenž zapouzdřuje všechny početní úkony ke tvorbě splatu (viz kapitola 2.3) a jeho následného vykreslení. Jelikož je nutné pro korektní zobrazení celého průhledného modelu mít splaty seřazené od nejvzdálenějšího k nejbližšímu rozhodl sem se do třídy CSplat umístit metodu pro výpočet a následné zjištění vzdálenosti od konkrétní transformační matice. Tato metoda by sice nemusela být zde, ale pro jednodušší manipulaci nad jednotlivými body je výhodnější takto umístěna.

3.2 Datová struktura pro popis celého modelu

Je reprezentována třídou CSplatArray. Přijímá již seřazené pole bodů, které následně rozděluje na vyšší množství polí o menším počtu bodů tak, že v jednotlivém poli zůstávají pouze úplně neprůhledné body, či body s jistou mírou průhlednosti. Při následném vykreslení se projdou všechny vzniklé pole bodů a omezí se tím jak testování změny stavového automatu OpenGL, tak i počet jeho změn. Jednotlivé pole bodů obsahují všechny informace

nutné k vykreslení všech bodů (pozice vrcholů, normálový vektor, barvu a koordináty textur) tak, aby bylo možné využít OpenGL funkci glDrawArrays a vyhnou se tak přílišnému volání funkcí pro definování každého vrcholu zvlášť (glVertex). Toto sice zvyšuje paměťovou náročnost celého systému, ale na druhou stranu podstatně zvyšuje rychlost vykreslování modelu.

3.3 Objekt připojitelný do grafu scény OSG

Jaj již bylo zmíněno v úvodu kapitoly je objekt reprezentován třídou CSplatPlane odvozenou od OSG::Drawable. Tudíž je připojitelný do grafu scény OSG jako vykreslitelný objekt. Zapouzdřuje vykreslování a řazení bodů celé kompozice bodů. Pro zobrazení více objektů do sebe tato třída musí mít schopnost připojení objektů stejného typu, při zachování korektního zobrazení. Proto se bude chovat jako strom (souvislý acyklický neorientovaný graf [8]), s tím že rodičovský uzel kopíruje do sebe ukazatele na jednotlivé body potomků. Z toho plyne, že uzel připojený do grafu scény OSG bude zahrnovat v jednom jediném poli informace o všech bodech celé kompozice modelů. Kořenový uzel celé kompozice modelů provede nad celým polem postupně trojí seřazení a vytvoří si tři datové struktury popsané v předešlé podkapitole tak, že každá ze struktur bude reprezentovat pohled z jiného směru (seshora, od přední strany a od levé strany). Posléze při samotném kreslení se vybere strana podle transformační matice a vykreslí se.

3.4 Testovací modely

K ověření korektní funkčnosti datové struktury popsané v předešlé podkapitole je potřeba vytvořit skupinu testovacích modelů, zahrnutých přímo v systému. Každý jednotlivý testovací model je odvozen od třídy CSplatPlane (viz. obrázek 3.1) tak, aby mohl být připojen do kompozice modelů a následně do grafu scény. Jelikož bod nemá konstantní velikost průhlednosti (viz. obrázek 2.8) je potřeba při generování libovolného 3D modelu dbát na korektní umístění surfelů do povrchu tělesa.

3.4.1 Kvádr

Pro generování bodů na povrchu kvádr využijeme toho, že kvádr se skládá ze šesti stěn, jenž každé dvě protilehlé stěny jsou rovnoběžné. Následně generované body umístíme do mřížky tak, aby přechody průhlednosti surfelů mezi jednotlivými částmi plochy byly co nejrovnoměrnější. Tento objekt je reprezentován třídou CCubePlane.

3.4.2 Koule

U generování bodů na povrchu koule budeme surfely umísťovat se stejným zřetelem jako u krychle. A pro určení normálového vektoru $\vec{N}$ každému surfelu využijeme vlastnosti středové symetrie koule. Pro každý bod A na povrchu koule a jeho normálový vektor $\vec{N}$ platí: $A - S = \vec{N}$, kde S je střed koule. Tento testovací objekt je reprezentován třídou CSplatSphere.

3.5 Načítání 3D modelů z externích zdrojů

Na rozdíl od objektů popsaných v předchozí kapitole, které jsou jen pro testovací účely, je načítání 3D modelů z externích zdrojů více zaměřeno na praktickou využitelnost systému. Běžné formáty souborů 3D modelů využívají polygonální síť, což je nutné převést na bodově orientovaný model. Postupně převedeme obecný polygon na trojúhelníky, které se posléze jednodušeji přetransformují na odpovídající počet surfelů. Tento objekt je reprezentován třídou CSplatPlaneFile.

3.5.1 Transformace trojúhelníku na surfely

Při transformaci z obecného trojúhelníku v prostoru budu vycházet z jeho velikosti. Pokud budu mít možnost ho aproximovat jediným surfelem o předem zadané velikosti, pak toho musím využít. V opačném případě přichází na řadu vyplnění obsahu jeho obsahu pomocí podobného principu, jako u testovacích objektů. Budu postupně procházet trojúhelníkem od nejdelší strany ke zbývajícímu bodu a po přímkách pokládat surfely.

3.6 Třída CSplat

Tato třída reprezentuje samostatně existující a vykreslitelnou entitu. Při svém vytvoření (volání konstruktoru) již plně reprezentuje surfel. Při volání konstruktoru se ze zadané pozice a normály, popřípadě velikosti či barvy ve formátu RGBA, vytvoří za použití rovnic popsaných v kapitole 2.3. Vrcholy rovnostranného trojúhelníku se uchovávají v jednorozměrném poli. Normálový vektor, barva a pozice jsou samostatně uloženy.

V této fázi se objekt po zavolání inline metody Render vykreslí podle aktuální transformační matice. Pro použití ve větším celku, kde bude potřeba řadit všechny předem vytvořené body byla implementována dvojice metod, které dokáží spočítat a následně přeposlat vzdálenost od transformační matice. Pro rychlejší získání vzdálenosti objekt tuto informaci uchovává až při opětovném volání metody k jejímu vypočtení provede výpočet.

Pro dynamickou změnu velikosti bodu jsem implementoval dvojici metod, jednu k nastavení nové velikosti a druhou k přepočítání vrcholů rovnostranného trojúhelníku. Objekt taktéž podporuje změnu barvy metodou SetColor a zjištěni stavu průhlednosti, která navrací kladnou hodnotu pokud nastavená barva má alfa kanál menší jedné.

3.7 Třída CSplatArray

Tato třída reprezentuje datová struktura pro vykreslování shluků bodů s možností vykreslení, a to metodou Render, do které vstupuje informace o směru vykreslování. Pro vytvoření této struktury, je jí nutné předat jednorozměrné pole entit popsaných v předešlé podkapitole. Z tohoto pole se vezmou postupně informace o vrcholech trojúhelníků a uloží se do jednorozměrného pole reálných čísel nazvaného Verticies. Podobným způsobem se zkopírují normálové vrcholy a informace o barvách. Narazí li se na změnu průhlednosti (předchozí byl průhledný a tento není, nebo opačně) vytvoří se nová struktura a pokračuje se stejným způsobem až do posledního zadaného bodu. Takto vytvořené struktury jsou uloženy ve STL vectoru nazvaného Ar.

Při zavolání metody Render s příslušným směrem vykreslování se postupně prochází STL vector Ar a za využití OpenGL funkcí glVertexPointer, glNormalPointer, glColorPointer, glTexCoordPointer, glDrawArrays se vykresluje podle aktuální transformační matice

celá scéna. Při průchodu tímto vectorem testuji první barva na hodnotu průhlednosti, a pokud určí daná hodnota, že jde o část průhlednou, vypínám Z–buffer a zapínám průhlednost. V opačném případě, jestliže jde o část neprůhlednou, vypínám průhlednost a zapínám testování hloubky.

3.8 Třída CSplatPlane

Tato třída odvozená od OSG::Drawable je připojitelná do grafu scény OSG, a pro své vykreslení využívá virtuální metodu drawImplementation podle doporučení manuálu OSG.

Do této třídy se postupně vkládají body, nebo objekty stejného typu, jako je tato třída. Při vložení bodu se uloží ukazatel na bod do jednorozměrného pole pro pozdější využití. Při vkládání celého modelu se zapamatuje ukazatel na takto vložený model a jeho body se přidají k bodům již existujícím. Po ukončení vkládání, ať bodů či celých modelů, vypočítám ohraničující kvádr (Bounding box) kolem výsledného modelu. K tomu aby byl objekt použitelný pro vykreslování je ještě nutné zavolat metodu nazvanou SortAll. Která postupně generuje tři transformační matice podle pohledu (seshora, od přední strany a od levé strany) a velikosti ohraničujícího kvádru. U každé takto vypočítané transformační matice prochází celé pole bodů a u každého zavolá metodu na vypočítání vzdálenosti. Posléze celé pole seřadí od nejvzdálenějšího prvku k nejbližšímu. S pomocí takto seřazeného pole vytvoří jeden prvek typu popsaného v předešlé podkapitole. V tomto bodě je Model připraven k vykreslení.

Při volání metody pro vykreslení se akorát rozhodne, které ze tří struktur pro vykreslení shluku bodů se bude volat metoda Render a ve kterém směru. Pro výběr směru a určí struktury slouží metoda Update, která je volána v update cyklu systému. Tato metoda zpětně projde zpětně celý graf scény OSG a zjistí aktuální transformační matici, podle které rozhodne co a v jakém směru se bude nadále vykreslovat.

3.9 Třída CCubePlane

Při vytváření kvádru je nutné znát jeho rozměry, a plošnou velikost bodu, jímž se bude plocha vyplňovat. Proto konstruktor tohoto objektu obsahuje trojsložkový vektor (osg::Vec3) rozměrů jednotlivých hran. Pro každou stěnu generuji body tak, aby byl zachován její celistvý povrch a vlastnosti popsané v kapitole 3.4.1.

3.10 Třída CSplatSphere

K vytvoření kole plně postačuje znalost jejího poloměru. Postupně si ji rozděluji na kružnice. Jednotlivé kružnici přiřazuji body po jejím obvodu tak, aby byly zachovány vlastnosti popsané v kapitole 3.4.1 a nevytvářely se body se stejným středem.

Kapitola 4

Implementace

V této kapitole se budu zabývat samotnou implementací systému, tak jak byl navržen v kapitole předešlé. První a stěžejní záležitostí je Třída samostatně vykreslitelné entity, posléze její příprava a napojení pro vykreslování. Následována generátory testovacích objektů a v uzavřena načítáním objektů z externích zdrojů.

4.1 Volba implementačního jazyku

Jako implementační jazyk jsem zvolil C++. C++ jako objektově orientovaný programovací jazyk je rozšířením jazyka C. C++ podporuje několik programovacích stylů (paradigmat) jako je procedurální programování, objektově orientované programování a generické programování, není tedy čistě jazykem objektovým. V současné době patří C++ mezi nejrozšířenější programovací jazyky.

Bjarne Stroustrup vyvinul C++ (původně pojmenované „C with Classes“) v roce 1983 v Bellových Laboratořích (Bell Labs at Murray Hill, New Jersey) jako rozšíření jazyka C. Rozšiřování začalo přidáním tříd, následováno dalšími rysy, virtuální funkce, přetěžování operátorů, vícenásobná dědičnost, šablony a zacházení s výjimkami.Standard programovacího jazyka C++ by ratifikován v roce 1998 jako ISO/IEC 14882:1998, stávající verze je z roku 2003 (ISO/IEC 14882:2003). Nejnovější verze standardu, neformálně nazvaná C++0x, je prozatím ve vývoji.

4.2 Volba 3D grafických nástrojů

Pro tvorbu systému jsem si vybral OpenSceneGraph, který je výkonným 3D grafickým nástrojem využívaným vývojovými pracovníky pro tvorbu vizuální simulace, her, virtuální reality, vědeckých zobrazení a modelování. OpenSceneGraph je taktéž open source a platformně nezávislý. Je postaven na konceptu SceneGraphu, zařizuje objektově orientovaný systém postavený nad OpenGL. Osvobozuje vývojového pracovníka od implementace a optimalizace nízkoúrovňových grafických operací, a poskytuje mnoho přídavných pomůcek pro zvýšení rychlosti návrhu grafických aplikací.[6]

4.2.1 OpenSceneGraph

OpenSceneGraph je zcela napsán v jazyce C++ a postaven nad OpenGL. Čímž umožňuje plné využití STL a návrhových vzorů. Poskytuje vývojovou knihovnu, která je volně dostupná a zaměřené na požadavky konečného uživatele. Síla OpenSceneGraphu je v jeho

výkonu, rozšiřitelnosti a přenositelnosti. OpenSceneGraph také podporuje jednoduchou změnu vykreslovacího procesu, jako je implementace vlastních vykreslitelných entit, čehož bude využíváno při vytváření systému pro vizualizaci 3D modelů reprezentovaných množinou povrchových bodů[2].

Jádro grafu scény zapouzdřuje funkcionalitu OpenGL zahrnující nejnovější rozšíření. Poskytuje vykreslovací optimalizace jako ořezávání a řazení, a celá sbírku přípojných knihoven s jejichž pomocí je možné vyvinout výkonou grafickou aplikaci velice rychle.

Pro čtení a zápis souboru dat databázová knihovna (osgDB) přidává podporu pro širokou variabilitu datových formátů. Pomocí mechanismu dynamicky zasunovatelných pluginů podporuje načítání různorodých 3D dat a obrázků.

Podporované formáty 3D dat: LightWave (.lwo), Alias Wavefront (.obj), Carbon Graphics GEO (.geo), 3D Studio MAX (.3ds), Peformer (.pfb), Quake Character Models (.md2), Direct X (.x), Designer Workshop (.dw) a AC3D (.ac) a vnitřní .osg ASCII formát.

Podporované formáty obrázků: .rgb, .gif, .jpg, .png, .tiff, .pic, .bmp, .tga

4.2.2 OpenGL

Jak již bylo popsáno v předchozí podkapitole OpenSceneGraph je postaven na OpenGL. OpenGL (*Open G*raphics *L*ibrary) se obvykle definuje jako jazykově a platformně nezávislá API (rozhraní pro programování aplikací[9]) pro psaní aplikací, které vytváří 3D (/2D) počítačovou grafiku. Rozhraní sestává s více než 250 různých volání funkcí. Může být využita k vykreslení komplexní trojrozměrné scény za pomocí jednoduchých primitiv. OpenGL bylo vyvinuto Silicon Graphics Inc. (SGI) v roce 1992 a je populární v herním průmyslu, kde konkuruje Direct3D na platformách Microsoft Windows. OpenGL je široce užívána v CAD, virtuální realitě, vědeckých vizualizacích, leteckých simulátorech a vývoji videoher.[7]

4.3 Načítání 3D modelů z externího zdroje

Načítání modelů z externího zdroje je reprezentováno třídou CSplatPlaneFile, která při volání konstruktoru tento soubor otevírá a načítá z něj data do instance třídy typu třídy osg::Group za pomoci knihovny osgDB. Takto načtená data mají stromovou strukturu.

Následně pokud je načtení kompletní, procházím graf a pro všechny geode uzly, jenž obsahují alespoň jeden prvek typu geometry, volám vlastní metodu AddGeometry. Tato metoda prochází samotná data 3D modelu a pro každý nalezený trojúhelník zavolá metodu AddTriangle s nalezenými vrcholy, normálami a barvou.

Metoda AddTriangle vymění pozice bodů tak, aby byla jasně zřetelná nejdelší strana trojúhelníku. Posléze prochází rovnoběžně s nejdelší hranou trojúhelník a generuje body na úsečce vzniklé průnikem přímky a dvou zbývajících hran. Pro všechny generované body v ploše vymezené trojúhelníkem používám, při jejich vytváření, normálový vektor tohoto trojúhelníku.

4.4 Vstupní bod aplikace

Vstupním bodem celého systému je funkce main. Na jejím začátku se prochází argumenty příkazové řádky. Vždy, po nalezeném uceleném celku argumentů buď pro generování implicitních modelů, nebo načítání z externího zdroje, se provádí jedna z alternativ vytváření objektů. Po bezchybném průchodu všech argumentů se volá metoda CSplatPlane::SortAll nad kořenovým uzlem celé kompozice modelů, pro její připravení k zobrazení. Následně se

provádí nastavení zobrazení, připojení dat do scény a hlavní programová smyčka. Po jejím ukončení (ukončení systému) se uvolňují zdroje a systém ukončí svůj životní cyklus.

4.5 Ovládání systému

V této kapitole se budu zabývat ovládáním systému. Od jeho spuštění až k ovládání výsledného zobrazení.

4.5.1 Spouštění aplikace

Systém je implementován jako konzolová aplikace. Ke svému spuštění vyžaduje zadání parametrů ke generování implicitních objektů, či načítání dat z externích zdrojů. Po korektním zpracování parametrů se provádí konečné úpravy celého zobrazovaného modelu (kompozice modelů) a řazení bodů pro vykreslení. Následně se inicializuje grafické prostředí. Parametry:

- -h

  Vypíše na standardní výstup text s nápovědou, jak korektně spouštět systém.

- -g

  Slouží ke generování objektů. Za tímto parametrem musí následovat informace o generovaném tvaru následně popsané.

  - box

    Tento parametr slouží pro generování krychle. Hned za tímto parametrem následují volitelná nastavení pro generování tvaru.

    * a=<hodnota> - velikost strany a kvádru
    * b=<hodnota> - velikost strany b kvádru
    * c=<hodnota> - velikost strany c kvádru
    * size=<hodnota> - velikost bodu, kterým se bude povrch reprezentovat
    * RGB <h h h> - RGB barva výsledného objektu
    * RGBA <h h h h> - RGBA barva výsledného objektu

  - sphere

    Tento parametr slouží pro generování koule. Hned za tímto parametrem následují volitelná nastavení pro generování tvaru.

    * r=<hodnota> - poloměr výsledné koule
    * size=<hodnota> - velikost bodu, kterým se bude povrch reprezentovat
    * RGB <h h h> - RGB barva výsledného objektu
    * RGBA <h h h h> - RGBA barva výsledného objektu

- -f

  Slouží pro načítání z externího zdroje. Za tímto parametrem musí následovat cesta a název souboru ke čtení. Volitelný parametr -size <hodnota> slouží k předefinování velikosti bodu, kterým se bude daný model reprezentovat

4.5.2 Ovládání scény

Pro manipulaci se scénou jsem využil původní nastavení OpenSceneGraphu. Pro manipulaci je používána myš. Rotace se provádějí stiskem levého tlačítka myši, kdy OSG podle její vybere střed rotace a následně provede rotaci kolem tohoto bodu. Pro hloubkové transformace respektive (přiblížení či oddálení) je použito pravého tlačítka myši a následného vertikálního posunu. K transformaci po zbývajících osách je využíváno středové tlačítko.

Důležité klávesy:

- S - zobrazí frame rate (počet snímků za jednotku času)
- L - zapíná/vypíná osvětlení
- T - zapíná/vypíná textury
- mezerník - uvede transformační matici scény do počátečního stavu
- Esc - ukončí aplikaci

Kapitola 5

Výsledky

Systém byl testován na: AMD Athlon(tm) XP 2200+ 1.80 Ghz, 768 MB RAM, grafická karta ATI RADEON 9200 SERIES 128MB AGP 8X, verze ovladače 8.252–060503a–038185C–ATI a dosahuje výsledků uvedených v tabulce 5.1.

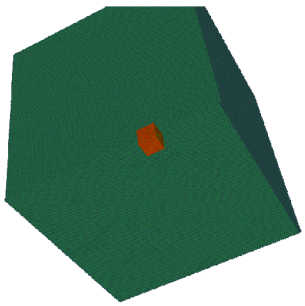
Název	celkový počet bodů	počet průhledných bodů	frame rate
Implicitní objekt obrázek 5.1	240 000	120 000	5.98
Implicitní objekt	240 000	0	7.20
Viper MK2 obrázek 5.2	389 703	0	5.03
Viper MK2 50% průhlednost	389 719	389 719	4.4
Postavička obrázek 5.3	1 037 917	0	1.4
Postavička obrázek 5.3	146 311	0	11.6
Sofa obrázek 5.4	183 153	0	10

Tabulka 5.1: Výsledky testování

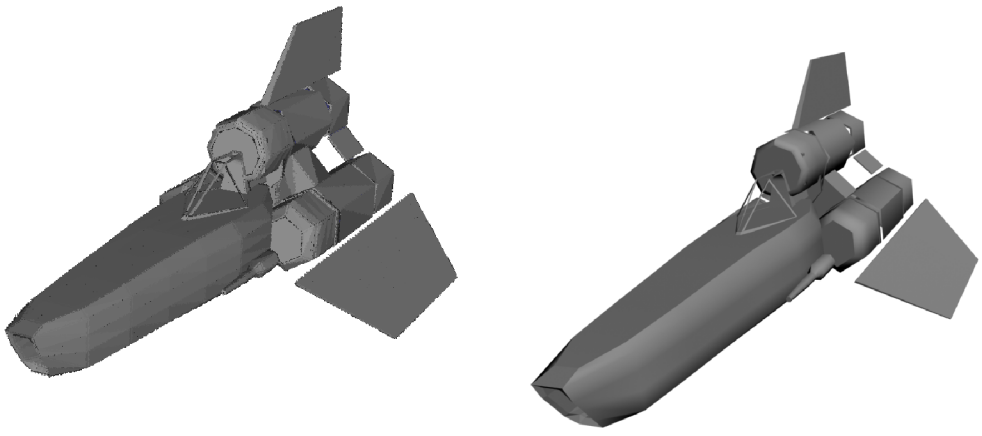
Název	počet polygonů	frame rate
Viper MK2 obrázek 5.2	1 924	180
Postavička obrázek 5.3	26 190	140
Sofa obrázek 5.4	60 988	60

Tabulka 5.2: Originální polygonální modely

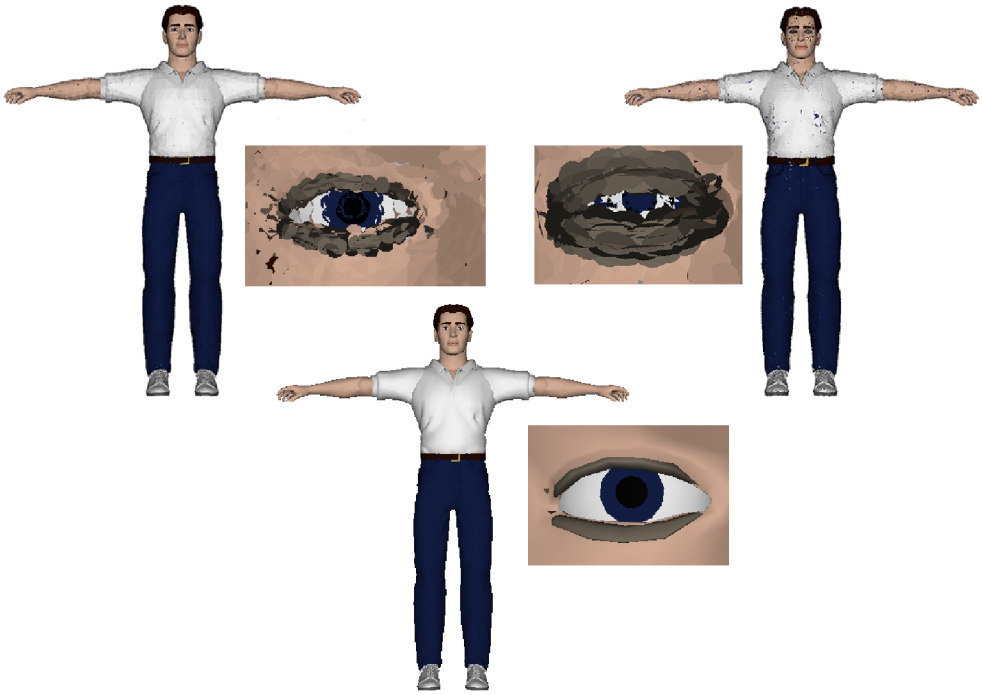
Z údajů v tabulce 5.1 je patrné, že při zachování stejného, či podobného množství bodů a změně poměru mezi neprůhlednými a průhlednými body dochází k podstatné změně rychlosti překreslení.



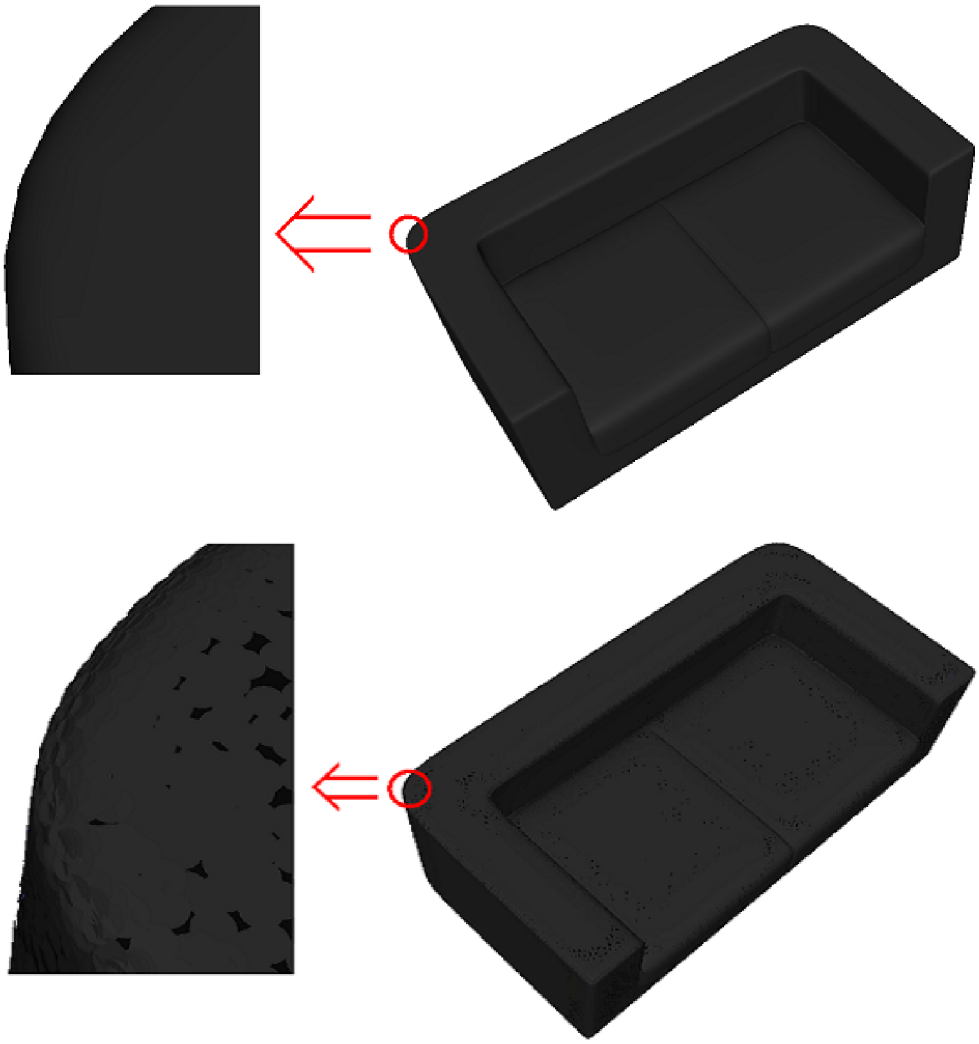
Obrázek 5.1: Implicitní model dvou krychlí vložených do sebe



Obrázek 5.2: Testovací model Viper MK2 na levo výstup z vyvíjeného systému, na pravo originální polygonální model



Obrázek 5.3: Testovací model postavičky z leva 1 037 917 bodů, 146 311 bodů a originální polygonální model a u každého detail levého oka



Obrázek 5.4: Testovací model postavičky sofa.3ds na hoře polygonální model dole výstup systému

Kapitola 6

Závěr

Práce popisuje systém pro vizualizaci 3D objektů reprezentovaných množinou povrchových bodů. V rámci zadání se podařilo vytvořit systém s možností komponování scény s více objektů, ať již implicitních (kole, kvádr), či v rámci OSG dostatečného množství a různorodosti podporovaných formátů pro načítání dat z externích souborů. Objektům je možné při jeho zpuštění určit velikost surfelu, kterým se bude dyný povrch popisovat, a taktéž i barvu jak ve formátu RGB, tak i RGBA, při zachování korektního zabrazení celé scény. Ovládání systému pomocí myši je velmi intuitivní. Přesto nemůže co do výkonu konkurovat profesionálním systémům stejného zaměření.

Z naměřených hodnot v tabulkách 5.1 a 5.2 a obrázku 5.3 vyplívá, že splatový model, může menším množstvím bodů obsáhnout kompletní topologii tělesa v určité vzdálenosti od pozorovatele, pro vyšší přiblížení je nutné měnit velikost a počet bodů objektu, tak aby byly zobrazitelné detaily. Naproti tomu reprezentaci polygonální stačí, aby v této mřížce nebyly mezery, což by mělo rušivý efekt. Popisováním topologie objektu za pomocí splatů je, možné tento objekt částečně vyhladit.

Systém může být nadále modifikován a vylepšován, a to především zvýšením výkonu při vykreslování za využití Vertex/Pixel shader, optimalizacím při generování a načítání objektů, či přídáním většího množství implicitních testovacích objektů.

Literatura

[1] Miguel Sainz, R. P.: Point-based rendering techniques. [online], [cit. 2007-05-6].
URL http://www.ifi.unizh.ch/vmml/admin/upload/Points_CAG.pdf

[2] OSG: OpenSceneGraph. [online], [cit. 2007-05-6].
URL http://www.openscenegraph.com/index.php

[3] V., C.: Počítačová grafika Modelování pevných těles. [online], [cit. 2007-05-10].
URL http://bimbo.fjfi.cvut.cz/~chalupec/pogr/system/files/pogr-12-modelovaniteles.pdf

[4] Weisstein, E. W.: NURBS Surface. From MathWorld–A Wolfram Web Resource. [online], [cit. 2007-05-6].
URL http://mathworld.wolfram.com/NURBSSurface.html

[5] Wikipedia: The Free Encyclopedia. [online], [cit. 2007-05-6].
URL http://en.wikipedia.org/wiki/Surfel

[6] Wikipedia: The Free Encyclopedia. [online], [cit. 2007-05-6].
URL http://en.wikipedia.org/wiki/OpenSceneGraph

[7] Wikipedia: The Free Encyclopedia. [online], [cit. 2007-05-6].
URL http://en.wikipedia.org/wiki/Opengl

[8] Wikipedie: Otevřená encyklopedie. [online], [cit. 2007-05-6].
URL http://cs.wikipedia.org/wiki/Strom_%28graf%29

[9] Wikipedie: Otevřená encyklopedie. [online], [cit. 2007-05-6].
URL http://cs.wikipedia.org/wiki/API

Seznam příloh

A CD, obsahující zdrojové kódy systému, programovou dokumentaci, elektronickou verzi práce, spustitelnou aplikaci